PSTAT 5LS Lab 3

Professor Miller

Week of April 24, 2023

Section 1

Learning Objectives

R Learning Objectives

1. Learn about reproducible randomness by "setting seeds"
2. Understand the basics of simulation in R

Statistical Learning Objectives

1. Understand how to translate a real data scenario into a simulation setup
2. Explore sample-to-sample variability by observing a distribution of simulated p-hat values.

Functions covered in this lab

1. `set.seed()`
2. `simulate_chance_model()`
3. `sum()`

Section 2

Lab Tutorial

Pseudo-Random Numbers

It turns out that humans are very bad at generating numbers randomly. Take a look at this word cloud of results from a survey of students in an introductory statistics course, in which students were asked to provide a random three-digit number. The larger a number is, the more often it showed up in the data.

As you can see, people tended to choose 123 a lot, or just typed the same number three times. If these numbers were *truly* random, we'd wouldn't see this – we'd see most three digit numbers in the data, and each would appear only a few times.

What is a Seed?

Computers are *much* better at randomness than humans are, but they're actually still quite bad. "Random" numbers generated by computers are meant to *look* and *seem* random, but, the sequences of numbers they generate are completely predictable if you know where that sequence starts. That starting point is called a **seed**.

Think of the seed as, well, a seed: if you know what kind of seed you plant, you know exactly what kind of flowers will grow. By setting a seed, you can predict the exact sequence of "random" numbers that will grow from it.

You can choose the seed that your R session will use with a function called `set.seed()`.

Setting a Seed

Run the `setSeed` code chunk. No output will occur, but it will set the seed.

```
set.seed(8362)
```

We chose the number 8362 arbitrarily.

If you execute the code chunk that contains the `set.seed()` code again, it will start your random number generator at the beginning of the sequence.

When you knit your document, it will also start your random number generator at the beginning of the sequence.

Simulating a Chance Model

In lecture, we are learning about how to simulate results using a chance model (that assumes the null hypothesis is true) to evaluate the probability of getting results like we saw in the data or something even more extreme.

Professor Miller's colleagues Nick Seewald and Elaine Hembree created a special package called `stats250sbi` that is available for install through GitHub. (SBI stands for "simulation-based inference.") Our lab projects will have this package preinstalled; you can find instructions on how to install it for a new RStudio project you start in JupyterHub or on your personal RStudio installation at the bottom of your notes document.

Let's take a look at our first SBI function, `simulate_chance_model()`.

simulate_chance_model()

In order to understand our new function, let's utilize the help feature.

```
?simulate_chance_model
```

You have a spot to run this code in the `tryit1` code chunk. Notice how the help feature pulls up on the bottom right corner to describe the arguments.

`simulate_chance_model()`

So the function takes on these arguments:

- `chanceSuccess`: a number between 0 and 1 representing the probability of observing a "success"
- `numDraws`: the number of times to draw a poker chip from the bag needed to complete one repetition of the simulation
- `numRepetitions`: the number of times to repeat the simulation process

Think of a "success" as answering yes to a particular question. For example, a "success" in the Doris and Buzz example occurs when Buzz pushes the correct button. Likewise, a "success" in the rock-paper-scissors example occurs when scissors are thrown.

We can now apply this to an example!

Simulation Example

Lee is a teacher at a local high school who wanted to assess whether or not dogs physically resemble their owners enough for people to be able to correctly match a dog to their owner better than if just guessing. Lee, who is also a dog owner, showed pictures of two dogs to her class of 26 students. One photo was of the teacher's dog (Yoda) and the other photo was of a dog the teacher had never met. The students were asked to guess which dog was actually the teacher's. If dogs do not physically resemble their owners, the students would get a correct match with probability 0.50, since the students would be equally likely to choose either dog. It turned out that 24 of the 26 students correctly picked out the teacher's dog.

Let's go through the entire procedure from start to finish.

Sample Proportion

What is $\hat{p}$, the observed (sample) proportion of correct guesses?

$\hat{p} = \frac{24}{26} = 0.923$

Hypotheses

What are the hypotheses to be tested? State the hypotheses using symbols. Be sure to define the parameter. You'll want to define both H_0 and H_a.

$H_0 : p = 0.5,$

$H_a : p > 0.5,$

where p represents the **proportion** of students who **correctly** guess which dog belongs to their teacher.

Setting Up the Simulation

Assuming the chance model. . .
One draw	*one student guessing which dog is their teacher's*
Blue poker chip	*the student guesses the dog correctly*
Yellow poker chip	*the student has an incorrect guess*
Chance of blue	*0.5*
One repetition	*26 student guesses*

The number of draws is the sample size of our study!

Small Number of Repetitions

Let's try this out for a small number of repetitions, say, 10, so that we can see what the output is from the function.

```
##  [1] 0.5000000 0.4230769 0.5384615 0.3846154 0.3846154 0.6153846 0.5384615
##  [8] 0.6923077 0.4230769 0.4230769
```

The output is a vector of the simulated proportions, the simulated $\hat{p}$ values, under the chance model specified.

If we are going to simulate 100 times, 1000 times, 10000 times, we don't want to see the long output! So in the future, we will assign the output to a variable name so that we don't have to read pages and pages of output.

NOTE: We reserve the right to deduct a point from your lab project should you print the contents of the entire simulation vector. Always check your knitted document before submitting!

Code to Simulate

Let's try this code out in the `tryit2` code chunk. to run the simulation 100 times.

```
sim1 <- simulate_chance_model(chanceSuccess = 0.5,
                              numDraws = 26,
                              numRepetitions = 100)
```

`sim1` should now be in your environment in the top right corner.

Notice that the first time you run this code, you get the **exact same values** we got! This is because we both set the same seed! If you run the code again, your values will change and will differ from ours. This is because the number of times that you run your code dictates where your random number generator is in the sequence of random numbers.

Pro Tip: Use your **knitted** document's output to view the first set of random numbers being generated!

Making a Visualization of the Simulation Results

So, if we assigned the output to a variable name, how do we view the results?

Make a histogram!

We will make a histogram of this *numeric* variable `sim1`. In our histogram, we will introduce a new argument called `xlim` that will allow us to set the minimum and maximum values on the x-axis. Since proportions are always between 0 and 1, setting these values as the minimum and maximum is a safe bet. Feel free to change these values at your discretion.

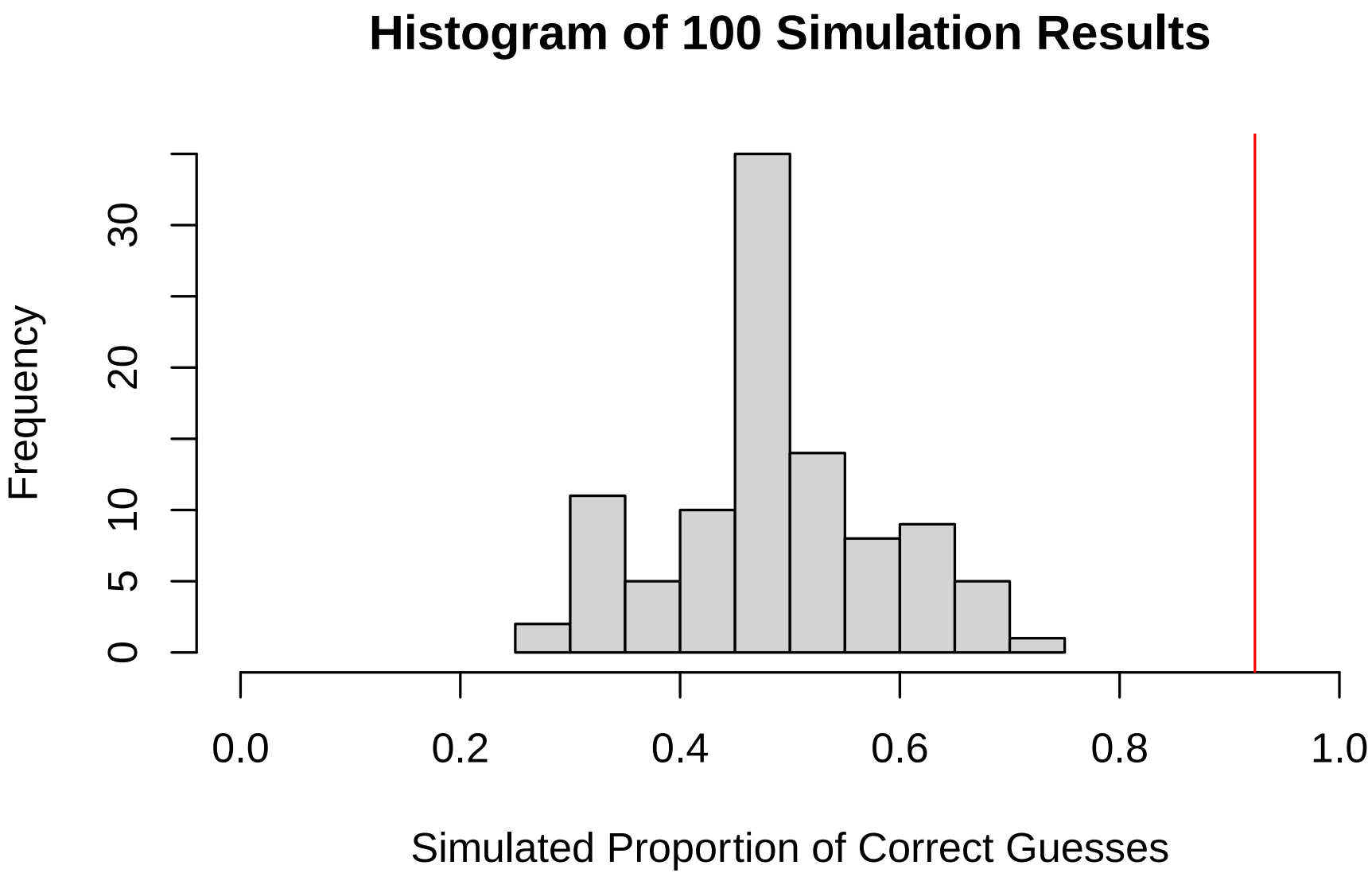
We will also make a vertical red line at the sample proportion. What was that value?

Histogram Code

Let's try this code out in the `tryit3` code chunk.

```
hist(sim1,
     main = "Histogram of 100 Simulation Results",
     xlab = "Simulated Proportion of Correct Guesses",
     xlim = c(0, 1))
abline(v = 24/26, col = "red")
```

Histogram of Simulation Results



Finding the Probability Of Getting This or More Extreme

As we learned in lecture, we are interested in the *probability that we will get our observed* $\hat{p}$ of 24/26 or a result that is even more extreme.

Since $H_a : p > 0.5$, the "as or more extreme" is actually **greater than or equal to**.

Let A represent something of interest to us. In this example, an observation in A is a simulated proportion of 24/26 or greater. The number of observations in A is the sum of the simulated proportions that are 24/26, 25/26, or 26/26. To find the probability of A in general, we use this formula:

$$p(A) = \frac{\text{number of observations of A}}{\text{total observations}}$$

Finding the Probability Of Getting This or More Extreme

So we will be:

1. looking for the number of observations that have a simulated proportion of 24/26 **or greater**
2. divide this number of observations by **100**, the number of times we ran the simulation

Code to Find this Probability

You can try this out in the `tryit4` code chunk.

```
sum(sim1 >= 24 / 26) / 100
```

```
## [1] 0
```

Let's break this down:

- **`sim1`** is the numeric variable representing a vector of the 100 simulation proportions
- the **`>=`** operator allows us to find things that are greater than or equal to 24/26 in the `sim1` variable
- the `sum()` function will add up the number of observations that meet this criteria. So here, it will find the number of observations of simulation proportions that are greater than or equal to 24/26
- divide the result of `sum()` by the total number of times the simulation was run

So What is the Probability?

```
## [1] 0
```

The probability of getting our observed sample proportion of 24/26 **or greater** is estimated to be 0.

What does this tell us about our observed sample proportion?

Is it *rare* or *not all that rare*?

Getting an estimated p-value of 0 is rare.

Conclusion

What do the results tell us about our research question?

Our estimated p-value of 0 is **very rare**. We will reject the null hypothesis at any reasonable significance level.

Do we have enough evidence to support the claim that dogs physically resemble their owners enough for people to be able to correctly match a dog to their owner **better** than if just guessing?

We **do** have enough evidence to support the claim that dogs physically resemble their owners enough for people to correctly match a dog to their owner better than if just guessing.